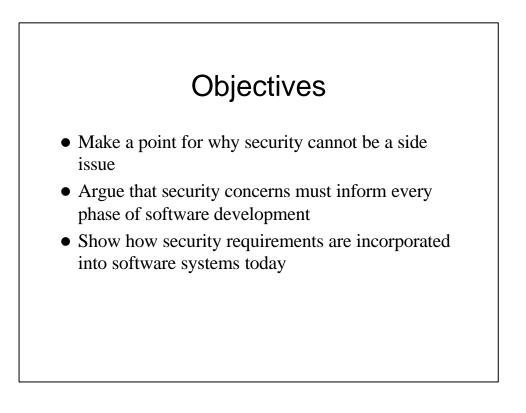
Software Engineering for Security

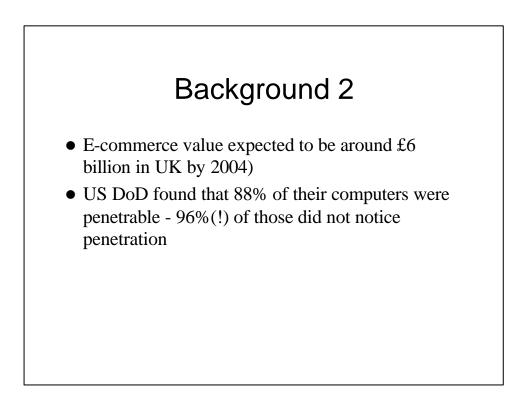
3C05 Advanced Software Engineering Unit ??

Daniel J. Hulme & Bruno Wassermann



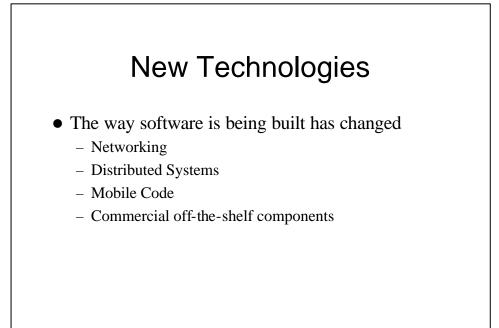
Background

- 60% of organizations have suffered security breach in the last two years
- only 37% of organizations undertake a risk assessment identifying critical assets
- 40% of companies that have experienced serious security breaches still do not have any contingency plans to deal with future attacks
- Source: Information Security Breaches Survey 2000, Technical Report, Department of Trade and Industry



Security System

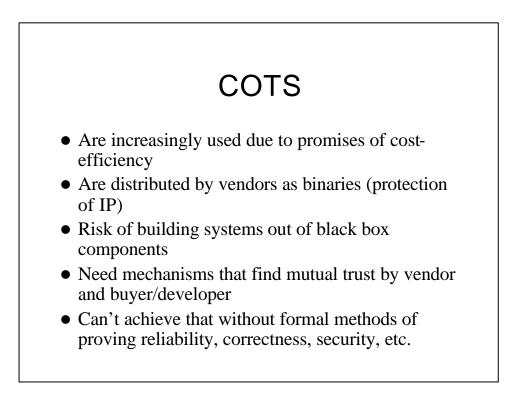
- A security system consists of *hardware* + *software* + people + procedures + culture
- Asset = something to protect
- Security Policy = mechanism to protect assets
- Vulnerability->Security attack->Security breach ->Compromise of confidentiality or integrity



Mobile Code

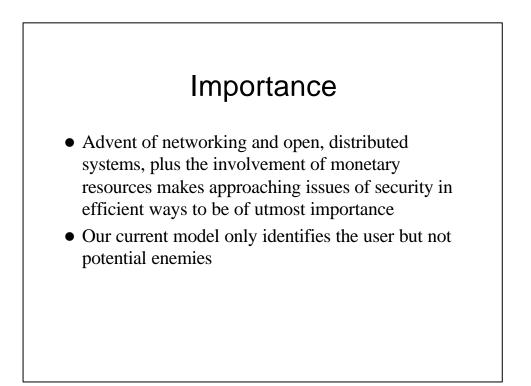
• Consider JAVA:

- Programs sent and received via networks
- Applets and RMI using object serialization
- Can execute system functions
- JAVA has a security architecture but it has (known) flaws
- Need to import and run such programs safely
- Protect users, programs and systems from each other



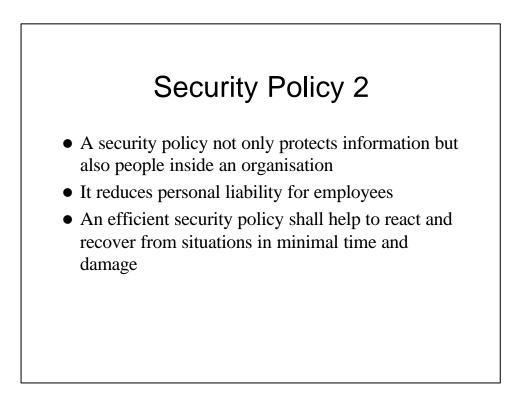
Distributed Systems

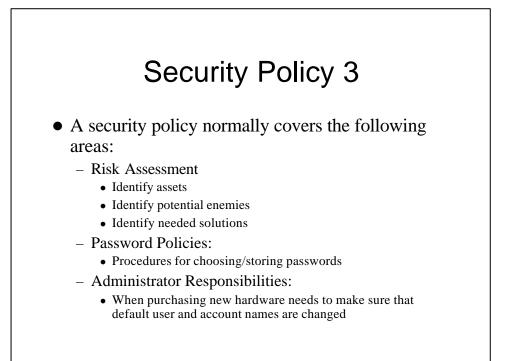
- Multiple organizations may co-operate via networked systems
- Each organization may use different platforms, security policies, procedures, and implementations
- Information about user permission may be held in different formats
- Dynamic population of objects with large variance in lifetime

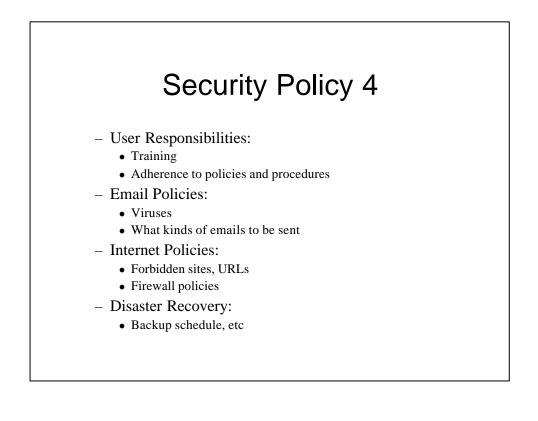


Security Policy

- The DoD Trusted Computer System Evaluation Criteria Glossary defines security policy as: "... the set of laws, rules and practices that regulate how an organization manages protects, and distributes sensitive information"
- Security policy establishes what must be done to protect information stored on electronic information systems
- Tells us "what" to do so that one can plan the "how"

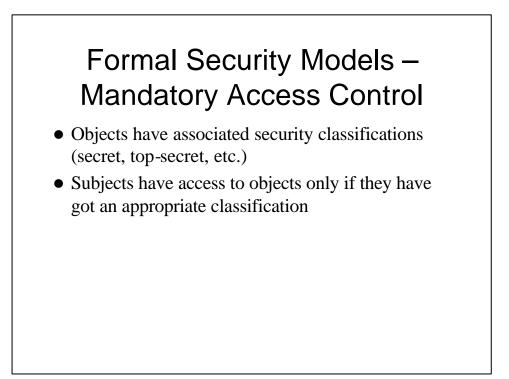






Security Requirements

- A security requirement is a detailed instantiation of a high-level organisational policy, I.e. detailed requirements of a specific system with respect to security policy
- Security requirements are non-functional requirement
- Often, security requirements come to light only after the functional ones have
- Often added as an afterthought to the system



Formal Security Models-Discretionary Access Control

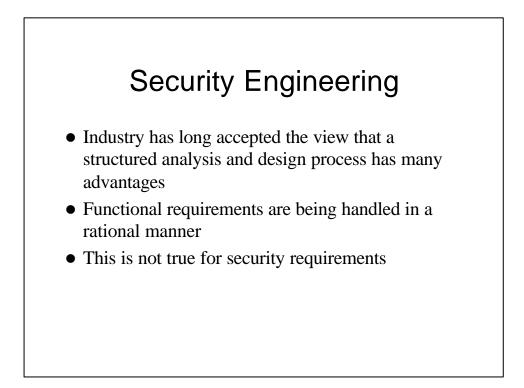
- Users belong to groups and/or processes
- Access restrictions based on identity of user
- User can pass access permissions to other users

Formal Security Models – Multilevel Security Model

- Each subject as well as object are assigned security level
- Objects can be read or written
- Subjects can only read objects at levels below them
- Subjects can write to objects at levels above them

Formal Security Models

- Multilevel security model enabled proof that information never trickled down the hierarchy
- All these formulations are clear and well-defined
- BUT
 - Access control works on a subject-object model
 - It considers the privileges of users and not of software
- The previous models are expressed in policy languages
 Check out: <u>www.camb.opengroup.org</u> ADAGE policy language)
- We want to integrate security requirements analysis with the already known standard requirements process



Security Requirements

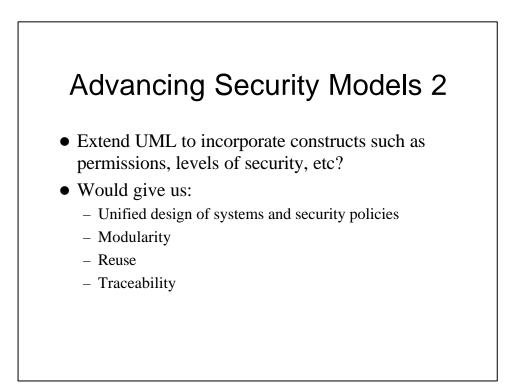
- Building a 100% secure system is hardly possible
- Would be very expensive
- Would inhibit users of the system carrying out their tasks
- Don't need to defend against all possible threats
- \rightarrow Adding security features consists of many compromises
- Planning for such features and adding them at a later point in the life cycle makes this task a lot more difficult
- Need to incorporate security requirements into our analysis and design

Unifying Security and System Models

- Tools that are used for requirements analysis and design are high-level OO models such as UML
- The business case drives requirements analysis
- Security modelling is still largely independent from standard modelling in practice

Advancing Security Models

- We need the same benefits for analysis of security requirements:
 - Requirements traceability
 - Automated analysis and reasoning
- We want engineering not craftsmanship
- Security requirements are "ilities" as found by other engineering disciplines





- A very serious problem is a mismatch between security frameworks in legacy systems and a target standard protocol
- The challenge here is to develop uniform policies and their implementation for a group of services that span different platforms

Legacy Security Mismatches Example

- CORBA
- Kerberos-based authentication
- Credentials (owned by CORBA client and each CORBA servant has its access control policy)
- UNIX
- User-password authentication
- File system uses access control(user, group, world)

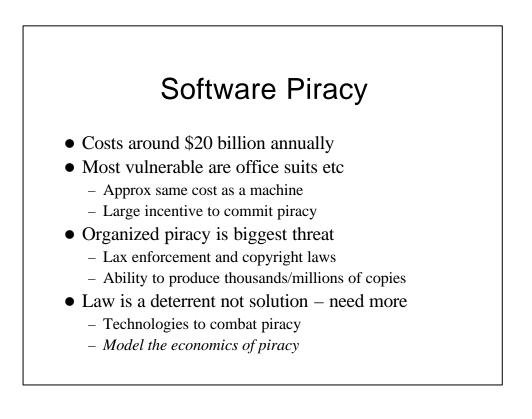
E.g. making services of a UNIX application A available via a CORBA object

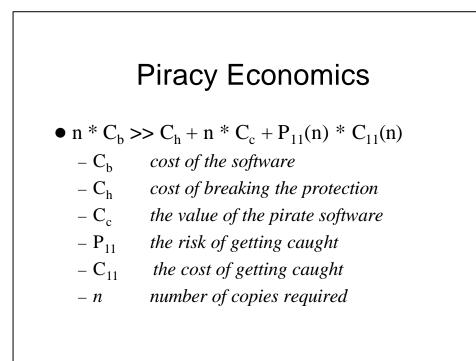
Now, if a particular login is not allowed to use A then the same user must not be allowed to invoke A's services through CORBA

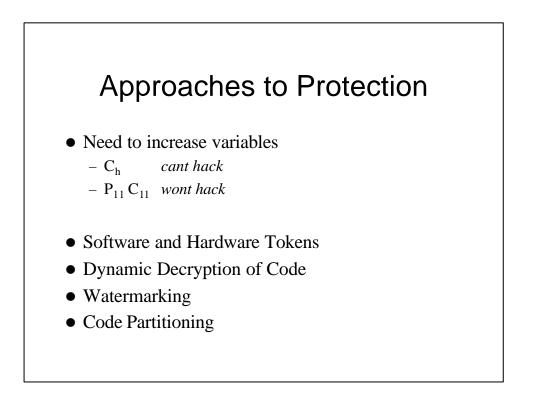
One Source of Mismatches

• Remember "ilities":

- Their implementation will be scattered throughout the code of the system
- Likely to find tangled code
- Problem is identifying these parts of the code, changing them and integrating changes back into the system
- Makes maintenance of security features a very difficult task







Software and Hardware Tokens

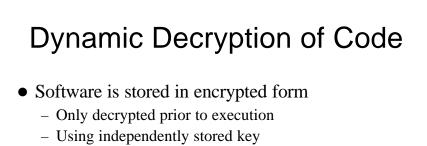
• SOFTWARE

- Licence file shipped with software
 - Most common technique
 - Checked every time software is run
- May include specific site information
 - E.g. network card address
- HARDWARE
- Physical 'dongle'
 - Attached too serial or parallel port
 - Software checks for token presence

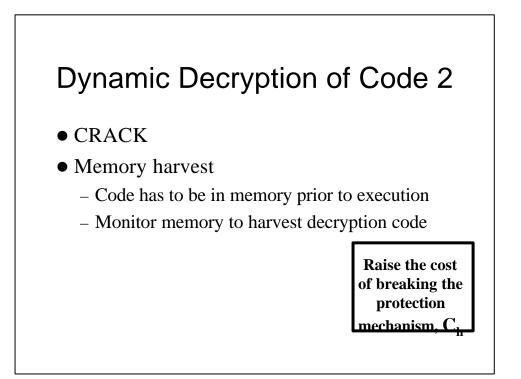
Raise the cost of breaking the protection mechanism, C_h

Software and Hardware Tokens 2

- CRACK
- Locate token-checking code an patch around it
 - Try not to use 'Licence' or 'Dongle' in your code
 - Use code debugger
- What about self-destructing code?
 - Use system-level debugger
 - Patch around tamper-resistant, self-checking and selfdestruct mechanisms

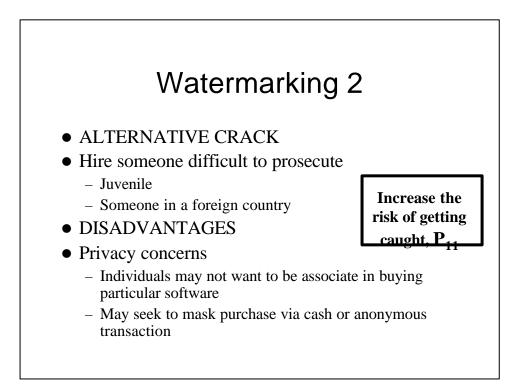


- Key could be associated with machine during manufacture
- Disadvantages
 - Unacceptable performance overhead
 - Would be difficult to legitimately move application from retired machine to new
- Not a common technique in industry



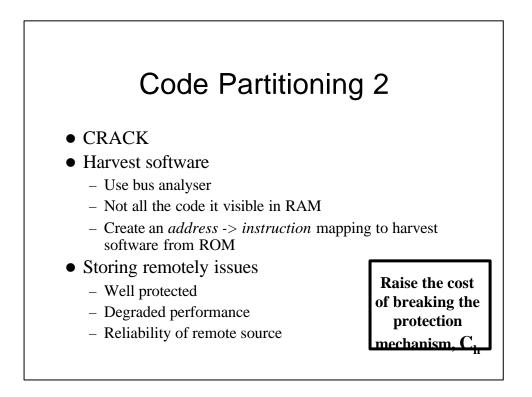
Watermarking

- Embed secret watermark in the software
 - Specific to the customer
 - Pirated copy can be traced back
- Stealth embedding
 - Difficult to find watermark
- Resilient embedding
 - Hard to tamper without damaging the media
- Static watermarks
 - A pattern in the program properties
- Dynamic watermarks
 - State activated "Easter eggs"



Code Partitioning

- Placing portion (substantial) of software in inaccessible memory
 - Partition in RAM and ROM
- Unfortunately has performance issues
- ROM is protected (but can be harvested)
- Also should protect processor and memory bus
 - More secure
 - Could store ROM partition remotely



Attacker Cost Models

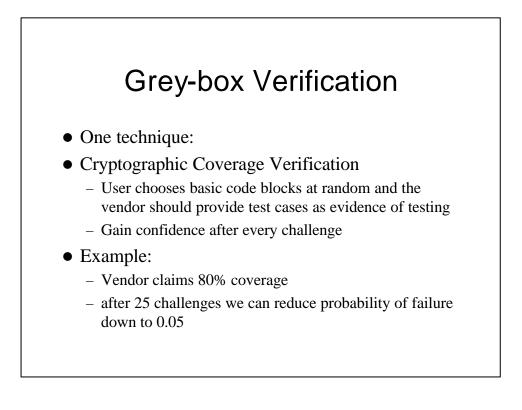
- There are still no accurate values on the cost of cracking software
- We need a better piracy economics model
- Attackers have full access to the hardware and software of the operating system
- Best solution would be to use tamper-resistant coprocessor executing partitioned software – but still can be hacked

Trusting Software Components

- Reference too: Judith & Raj Safety
- Software development today integrate COTS
 - Fraught with safety and security risk
 - Vendors may be unwilling to provide propriety information
- Vendors have two choices:
 - Black-box approach
 - Grey-box verification

Black-box Approaches

- Two approaches for user confidence:
 - In suit testing
 - Makes sure the components don't misbehave
 - System testing
 - Makes sure the system doesn't misbehave even if the components do
- Both require extensive testing
- Vendor does not have to disclose any intellectual property



Conclusions

- Intrusion detection community (CERT) deals with the status quo
- Don't come up with new designs or architectures
- We need to incorporate security engineering into standard analysis and design process
- Must not leave security requirements to be dealt with as a side-issue, an afterthought

